Principles for a Judgement Editor Based on Binary Decision Diagrams

Guillaume Aucher, Jean Berbinau, Marie-Laure Morin

Principles for a Judgement Editor Based on Binary Decision Diagrams

Guillaume Aucher
Univ Rennes, CNRS, IRISA
guillaume.aucher@irisa.fr

Jean Berbinau
French Association of IT Judicial Experts
(ex- & honorary member)
jean.berbinau@m4x.org

Marie-Laure Morin
CNRS, Cour de cassation (Former trial judge)
mlmorin@orange.fr

Vol. 6 No. 5 2019
Journal of Applied Logics — IfCoLog Journal of Logics and their Applications

Abstract

We introduce the theoretical principles that underlie the design of a software tool which could be used by judges for making decisions about litigations and for writing judgements. The tool is based on Binary Decision Diagrams (BDD), which are graphical representations of truth–valued functions associated to propositional formulas. Given a type of litigation, the tool asks questions to the judge; each question is represented by a propositional atom. Their answers, true or false, allow to evaluate the truth value of the formula which encodes the overall recommendation of the software about the litigation. Our approach combines some sort of 'theoretical' or 'legal' reasoning dealing with the core of the litigation itself together with some sort of 'procedural' reasoning dealing with the protocol that has to be followed by the judge during the trial: some questions must necessarily be examined and sometimes in a specific order. That is why we consider extensions of BDD called Multi-BDD. They are BDD with multiple roots corresponding to the different specific issues that must necessarily be addressed by the judge during the trial. We illustrate our ideas on a case study dealing with French trade union elections which has been used throughout our project with the Cour de cassation. We also introduce the prototype developed during our project and a link with restricted access to try it out.

1 Introduction

The systematic form of many legal systems derives from Roman law. Romans of antiquity were the first to integrate and apply the methods and rigor of Greek philosophy and logic to the law, especially Gallius, Mucius Scaevola in the 2^{nd} century BC and subsequent jurisconsults of the imperial period. The rediscovery of Roman law via Justinien's compilations in the 12^{th}–13^{th} century in northern Italy and France followed by its interaction and confrontation with the humanists of the 15^{th}–16^{th} century rang the knell of custom and unwritten law and greatly contributed to shape the landscape of law in continental Europe [39]. It is therefore not surprising that the structures of Roman law and of many subsequent and current legal systems are closely related to logic and mathematical theories. Over the centuries, many jurists have brought to the fore the systematic and deductive power of legal systems, such as for instance the Roman Cicero (106–43 BC, "*jus in artem redigere*"), the French Domat (1625–1696), the German Leibniz (1646–1716) or the Italian Beccaria (1738–1794).

Logic, sometimes viewed as the "calculus of computer science" [28, 42], then became a natural theoretical background to address new kinds of legal matters. In the new societal and virtual context raised by the recent technological developments of internet and computers, law must still be enforced, transparent, accountable and

understandable by anybody more than ever. As it turns out, an important amount of works at the intersection of logic, law and artificial intelligence has emerged over the last decades. We are now going to briefly summarize it and we refer the reader to the handbooks [22, 18] or survey articles [15, 35, 27] for more details and pointers.

1.1 A status quo in logic and law

Originally, logic was intended to be used for the representation of law in a clear and unambiguous manner. On top of this representation, some kind of reasoning could then take place to infer some information. Sergot and Kowalski [37] were pionneers in that field, with the use of logic programming that they applied to the formalization of the British Nationality Act. However, they encountered difficulties with the Prolog treatment of negation as failure and with the lack of deontic operators [29]. In Prolog, something is said to be false if it is not known or cannot be inferred to be true. Hence, if a program cannot show that an infant was born in the UK, it assumes that it was not.[1] More generally, researchers realized that several aspects of the law which can not be dealt with standard (Fregean) logic had to be taken into account, such as the need to handle exceptions, conflicting rules, vagueness, open texture (i.e. the failure of natural languages to determine future usage of terms), counterfactual conditionals and the possibility of rational disagreement. Some of these peculiarities of legal reasoning were and are in fact still addressed by various non–classical logics in an area sometimes called 'applied logic' [32].

Representation of law. Among the logical formalisms used to represent law, deontic logic provides formal tools for the clarification of the meaning of normative terms such as 'may', 'must' and 'shall', which play a central role in specifying the legal relations between agents (which can be human beings or machines). Therefore, it has been used in the analysis of law [29] and in the area of automated contract management. Other notions which play an important role in law and legal reasoning were also analysed logically, such as the notion of power, as involved in sentences such as "the president has the power to declare a state of emergency". Normative systems propounded by Alchourrón and Bulygin were another influential approach to represent legal systems [2]. A normative system is a set of norms, which are pairs of the form $\langle condition, consequence \rangle$: if the condition holds then the consequent *must* hold. Unlike formulas of deontic logic, they do not bear truth values.

Several structural features of legal regulations, such as the use of exception, the use of hierarchies of legislation to resolve conflicts, cross references to other

[1] This thread of research in legal reasoning using logic programming is in fact still active, as witnessed for example by the Japanese PROLEG system [36].

parts of the legislation, deeming provisions, conditions under which the legislation is applicable, and conditions for the validity of particular norms, led to the use of non-monotonic logics [34]. These are logics where the consequence relation is not monotonic, meaning that adding a formula to a theory does not necessarily produce an increase of its set of consequences. However, these formalisms proved inadequate to provide a general means of conflict resolution. These conflicts are sometimes due to conflicting interpretations of the law and get even more difficult to handle in the context of *stare decisis* (a legal principle of Common Law by which judges are obliged to respect the precedents established by prior decisions of higher jurisdictions). These difficulties led to a shift of focus from the modes of representation to the modes of reasoning.

Reasoning about law. Law and its practice are subject to different kinds of reasoning:

- *Case-based reasoning* uses considerations about precedent legal cases to show how they justify particular outcomes in a new case (following the *stare decisis* principle of the Anglo-American Common Law). The problem is then to map appropriately the precedent cases to the new case. Several models have been developped and logically formalized, notably by McCarty [31]. This problem involves the classification of the facts of a case under legal concepts and the interpretation of these legal concepts.

- *Practical and teleological reasoning* deals with the reasoning involved in the justification of choices that the arbiter has to make in some legal decisions. These justifications should be in line with the underlying purpose of the law. This involves to be able to derive normative consequences from the classification of facts and the interpretation of legal concepts.

- *Evidential reasoning* is the kind of reasoning that occurs when judges strive to establish the facts on the basis of evidences.

Different kinds of logical formalisms were developped to address these different kinds of reasonings, and in particular numerous works resorted to argumentation theory. However, legal reasoning mostly arises in the conduct of a dispute which is regulated by a particular procedure. The outcome of this dispute does not only depend on facts and a body of law, but also on the procedure itself: whether it is a criminal proceedings or a civil proceedings, to which party is assigned the burden of proof in this procedure, etc. A number of dialogue games models of legal procedure have been produced in the last 20 years [33].

All this said, a striking particularity of most of the works which have been pursued at the interface of logic and law in the last decades is that they were mostly driven by theoretical considerations and without much interaction with jurists and lawyers. Arguably, this work did not really catch the attention of the lawyers and jurists. In particular, they did not change the way they work or their actual practice of the law, except maybe for the adoption of large and online databases such as LexisNexis or Legifrance[2] (based on standards for legal documents such as LegalRuleXML) and the use of knowledge management systems [17, 16, 1]. This theoretical work did not seem for jurists to answer an actual need and it was somehow remote from their daily preoccupations, although the researchers could sense the potential and the important applicability of their work in the practice of the law.

1.2 Current problems in the application of law in France

The joint work reported in this article stems from actual problems in the application of law in France that were expressed to us by jurists and magistrates of the French Cour de cassation.[3] These problems are in fact not specific to France. The application of law is plagued with a series of problems which are difficult to overcome with the standard and present methods employed by jurists [30, 11].[4] First, the increasing diversity and complexity of legal texts and jurisprudence makes the work of jurists (and lawyers) very difficult to pursue nowadays. This complexity appears not only at the local or national level but is sometimes worsened by its interaction with the European level, and sometimes even the international level. Second, legal texts and jurisprudence are changing at a high pace in some areas and it is difficult for jurists (and lawyers) to cope and keep up-to-date with the current legislation and regulations. Third, there is a lack of consistency in the application of law, depending on the geographical part in which trials take place, on the local customs and sometimes the personnality of the judges, and more generally on the specific

[2] See `www.lexisnexis.com` and `www.legifrance.gouv.fr`.

[3] The main role of the Cour de cassation is to check that the law is applied correctly and uniformly in France mainly from a legal and procedural point of view, the determination of facts being left to the 'tribunaux de grande instance et d'instance' and 'cour d'appel'. Stemming from the 'justice retenue' dispensed by the French kings from the 13^{th} century on and formerly called 'Conseil des parties' and then 'Tribunal de cassation' during the French revolution, the Cour de cassation is one of the oldest juridical institution in France. It "is the highest Court in the French judiciary. [...] the Court of Cassation is thus required to find whether the rules of law have been correctly applied by the lower courts based on the facts. [...] If the decision of the lower court is quashed [cassée in French], the case has consequently to be heard again." (`www.courdecassation.fr/about_the_court_9256.html`)

[4] In some countries subject to common law, like the United Kingdom, such problems are worsen by the fact that some legal decisions are not made by professional jurists [11].

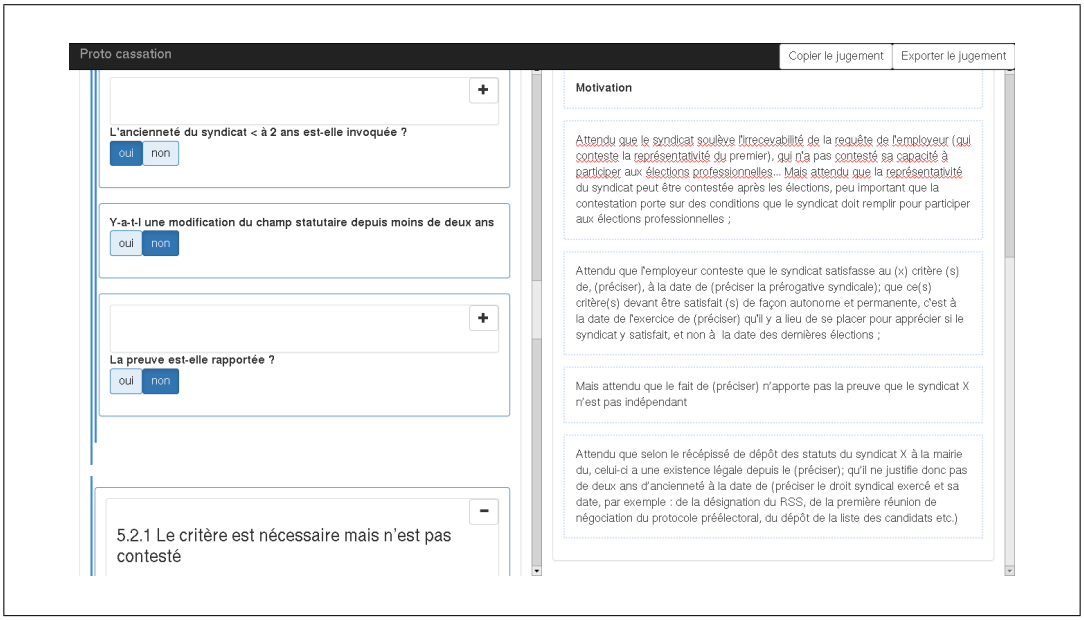


Figure 1: Screenshot of the graphical user interface.

political or social context in which a legal decision is taken.

The third problem is not novel and was already raised before the French revolution of 1789 by philosophers such as Voltaire, Diderot or Rousseau and the lawyer Linguet for example. It led to the vast enterprise of codification of the law. During the French revolution, the Assemblée voted for a Code penal inspired by ideas of Montesquieu and Beccaria and a series of 'revolutionary' projects led by Cambacérès were submitted. One had to wait for Napoleon and more peaceful times for the promulgation of the first comprehensive Code civil and other codes. The Napoleonian codification had a lasting impact over France and many other countries which adopted (partly) the French codes or took inspiration from them. However, as already noted by Thireau in 2009, "with the inflation of legal texts and regulations, the time of large codifications seems over" [39, p. 335]. Altogether, these three problems call for a new kind of solution.

1.3 A new kind of solution: a software assistant

The solution that we propose is to make use of a software, whose ultimate role is to help judges write a judgement and take better and well-informed decisions thanks to a series of questions to which she/he has to answer. These questions would be

backed up by the corresponding legal texts and jurisprudence.

This software assistant would indeed be a solution to the problems mentioned above. First, it would unify and uniformize the application of law in France: the kind of reasoning proposed by the software to sort out a given (type of) litigation could then be controlled and it could also be the same in every jurisdiction of France. Second, like any software, it could take into account the evolution of legal texts and jurisprudence to update the different kinds of reasoning and therefore cope with the increasing complexity of law. Third, its easy access to a large and up-to-date knowledge base comprising the current legal texts and jurisprudence would increase the chance for the judge to make well–informed decisions.

The graphical user interface (GUI) of this software is depicted in Figure 1. On the left hand side of the GUI, a guide for reasoning consisting of a series of questions is displayed. These questions and their underlying reasoning are backed up by legal texts and jurisprudence to which the user can have access whenever she/he wants. On the right hand side of the GUI, a judgement is produced automatically as the user answers the questions. The user can modify at any time the judgement produced and can also have an alternative graphical representation of the web of questions to which she/he has to answer on the left hand side.

1.4 Structure of the article

In Section 2, we recall the basics of propositional logic and BDD. In Section 3, we extend propositional logic with the examination operator $!\varphi$ and we provide a semantics for this operator based on Multi-OBDD. In Section 4, we consider as case study the problems of determining whether an association of employees in a firm can indeed be considered ('qualified') as a trade union. In Section 5, we show how the various algorithms that have been designed for OBDD can be used to solve and address specific kinds of legal issues that arise in the practice of the law. In Section 6, we describe the prototype that we have implemented during our project and provide a link with restricted access to try it out. We conclude in Section 7.

2 Propositional logic and BDD

In this section, we recall the basics of propositional logic and Binary Decision Diagrams (BDD for short, [20, 21]) and we show how they are related to each other. BDD provide a graphical semantics for formulas of propositional logic in which the truth–values of their atoms can be assigned in a specific order. This feature will play a role in the legal context since it will allow us to represent the *procedural* aspect of the practice of law (during a trial especially).

$\mathcal{I}(\varphi)$	$\mathcal{I}(\psi)$	$\mathcal{I}(\neg\varphi)$	$\mathcal{I}(\varphi \wedge \psi)$	$\mathcal{I}(\varphi \rightarrow \psi)$	$\mathcal{I}(\varphi \vee \psi)$
T	T	F	T	T	T
T	F	F	F	F	T
F	T	T	F	T	T
F	F	T	F	T	F

Figure 2: Semantics of propositional connectives.

2.1 Propositional logic

In the sequel, $\mathbb{P}$ is a set of *atoms* (propositional letters) denoted $p, q, r, \ldots$ and T and F are two symbols called *truth values* standing for True and False.

Definition 1 (Propositional language $\mathcal{L}$)**.** The language $\mathcal{L}$ is the set that contains $\mathbb{P}$ and such that

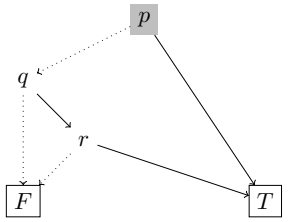
- if $\varphi, \psi \in \mathcal{L}$, then $\neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi) \in \mathcal{L}$;
- $\mathcal{L}$ contains no more formulas.

We introduce the following abbreviations: $\varphi \leftrightarrow \psi \triangleq (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi), \top \triangleq (p \vee \neg p), \bot \triangleq (p \wedge \neg p)$ for some $p \in \mathbb{P}$. The formula $\varphi[p \backslash \psi]$ denotes the formula φ where the atom p is uniformly substituted with ψ.

The intuitive reading of the formulas is as follows: $\neg\varphi$: "φ does not hold"; $\varphi \wedge \psi$: "φ holds and ψ holds"; $\varphi \vee \psi$: "φ holds or ψ holds"; $\varphi \rightarrow \psi$: "If φ holds then ψ holds".

Definition 2 (Interpretation)**.** A *total (partial) interpretation* is a total (resp. partial) function $\mathcal{I} : \mathbb{P} \mapsto \{T, F\}$ that assigns one of the *truth values* T or F to *every* (resp. *some* of the) atom(s) in $\mathbb{P}$. The set of total interpretations is denoted $\mathcal{C}$ and the set of partial interpretations is denoted $\mathcal{C}^p$. Note that $\mathcal{C} \subseteq \mathcal{C}^p$. If $\mathcal{I} \in \mathcal{C}^p$, then $Ext(\mathcal{I})$ is the set of *total* interpretations extending the interpretation $\mathcal{I}$, that is, for all $\mathcal{I}' \in Ext(\mathcal{I})$, for all $p \in \mathbb{P}$ such that $\mathcal{I}(p)$ is defined, we have that $\mathcal{I}(p) = \mathcal{I}'(p)$.

We can extend the domain of an interpretation function from the set of atoms to the set of all formulas of $\mathcal{L}$. This extension is inductively defined by the truth table given in Figure 2. If E is a set of interpretations, we say that a formula φ of $\mathcal{L}$ is *valid on* E when for all $\mathcal{I} \in E$, we have that $\mathcal{I}(\varphi) = T$. When $E = \mathcal{C}$, we simply say that φ is *valid*.



$\mathcal{I}(p)$	$\mathcal{I}(q)$	$\mathcal{I}(r)$	$\mathcal{I}(p \vee (q \wedge r))$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	F

Figure 3: OBDD and truth table of the formula $(p \vee (q \wedge r))$.

2.2 Binary Decision Diagrams (BDD)

Binary decision diagrams are graphical data structures for representing compactly the semantics of formulas of propositional logics. They have been widely used in the industry for the verification of computer hardware. Our presentation is adapted from the book of Ben-Ari [14], based on the articles on BDDs by Bryant [20, 21].

Definition 3 (Binary Decision Diagram, BDD)**.** A *binary decision diagram (BDD)* is a directed acyclic graph with a unique root, which is also called the *entry point*. Each leaf is labeled with one of the truth values T or F. Each interior node is labeled with an atom and has two outgoing edges: one, the *false edge*, is denoted by a dotted line, while the other, the *true edge*, is denoted by a solid line. No atom appears more than once in a branch from the root to a leaf.

During a trial, some questions have to be examined in a certain temporal order. This temporal order does not play a role from a logical point of view, in the sense that the truth value of a given statement will not depend on the order in which its arguments are examined. However, this temporal order plays a role from a procedural point of view when the judge constructs its judgment while answering the different questions.

This ordering is made explicit in *ordered* binary decision diagrams (OBDD): we can canonically associate to each OBDD an ordering corresponding to the order in which the different questions should be examined by the judge. However, these orderings of the different branches of the OBDD should be somehow 'compatible'.

Definition 4 (Compatible orderings)**.** Let $<^1, \ldots, <^n$ be orderings on $\mathbb{P}$, that is, for each $i \in \{1, \ldots, n\}$, $<^i$ is a total relation on a subset $\mathbb{P}^i \subseteq \mathbb{P}$. We say that $<^1, \ldots, <^n$

Input: A BDD *bdd*.

Output: A reduced BDD denoted `Reduce`(*bdd*).

Perform a recursive traversal of the BDD:

- If *bdd* has more than two leaves T and F, remove duplicate leaves. Direct all edges that pointed to a removed leaf to the remaining respective leaf.
- Perform the following steps as long as possible:
  1. If all outgoing edges of a node labeled p point at the same node labeled q, delete this node for p and direct p's incoming edges to q.
  2. If two nodes labeled p (distinct from roots) are the roots of identical sub-BDDs, delete one sub-BDD and direct its incoming edges to the other node.

Figure 4: Schematic algorithm `Reduce`.

are *compatible orderings* when for all $i \neq j$, there are no atoms $p, p' \in \mathbb{P}^i \cap \mathbb{P}^j$ such that $p <^i p'$ while $p' <^j p$.

Definition 5 (Ordered Binary Decision Diagrams, OBDD)**.** An *ordered binary decision diagram (OBDD)* is a BDD such that the orderings of atoms $<^1, \ldots, <^n$ defined by the branches stemming from the root are compatible. The *ordering associated to the OBDD* is $<^1 \cup \ldots \cup <^n$.

Example 1. Figure 3 shows an OBDD and the truth table of the formula $(p \vee (q \wedge r))$. The OBDD representation is more compact and avoids redundancy. One can indeed notice that the four rows of the truth table where p is true make the formula $(p \vee (q \wedge r))$ true, regardless of the truth values of q and r. This redundancy is represented in the OBDD by setting a true edge from the entry point p to the leaf labeled T.

Definition 6 (Operation `Reduce` on BDD)**.** The operation `Reduce` on BDD is defined in Figure 4.

Example 2. Figure 5 shows the application of a sequence of `Reduce` on an OBDD equivalent to the OBDD of Figure 3. First, we merge all the leaves labeled by F in one single leaf and we merge all the leaves labeled by T in one single leaf. Then we 'bypass' the r-node and the two r-nodes on the right because all outgoing edges lead to the same node. In the last step, we 'bypass' the q-node on the right.

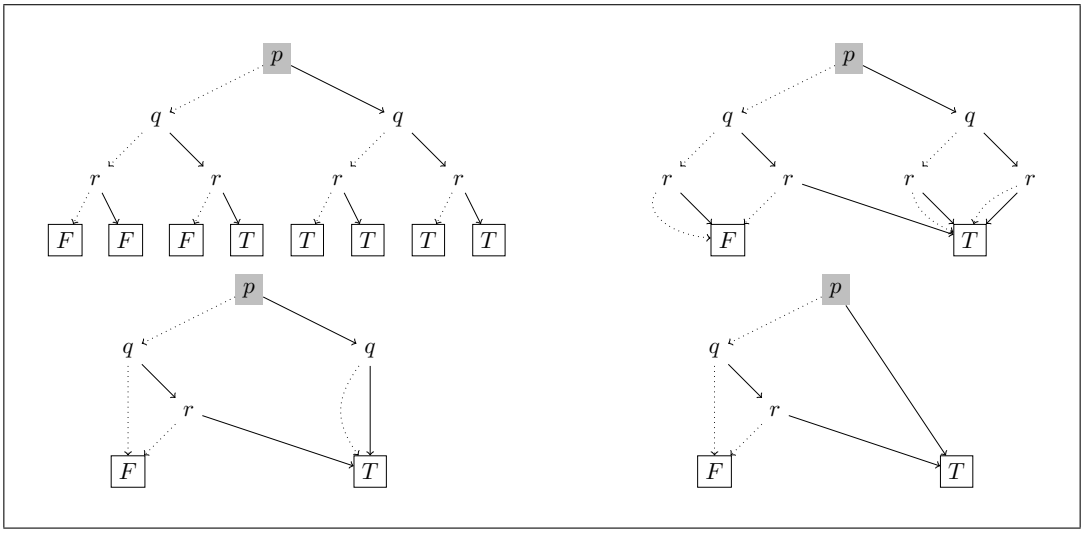


Figure 5: Step-by-step of algorithm `Reduce`.

Definition 7 (Operation `Apply` on OBDD)**.** The operation `Apply` on OBDD is defined in Figure 6.

Instead of the set-theoretical semantics of propositional logic based on the notion of interpretation, we can provide a semantics to propositional logic in terms of OBDD. The meaning of a propositional formula is completely determined by the OBDD associated to that formula, which is itself built inductively with the `Apply` Algorithm of Figure 6. The soundness of `Apply` is ensured by the following theorem:

Theorem 1 (Shannon expansion)**.** *For all formulas $\varphi, \psi \in \mathcal{L}_M$, for all $\star \in \{\wedge, \vee, \rightarrow\}$, the following formula is valid:*

$$\varphi \star \psi \leftrightarrow (p \wedge (\varphi[p\backslash\top] \star \psi[p\backslash\top])) \vee (\neg p \wedge (\varphi[p\backslash\bot] \star \psi[p\backslash\bot]))$$

For example, $\varphi \wedge \psi \leftrightarrow (p \wedge (\varphi[p\backslash\top] \wedge \psi[p\backslash\top])) \vee (\neg p \wedge (\varphi[p\backslash\bot] \wedge \psi[p\backslash\bot]))$.

Definition 8 (OBDD associated to a formula and an ordering)**.** Let $\chi \in \mathcal{L}$ and let $<$ be an ordering on the set of atoms occuring in χ. The OBDD associated to χ and $<$, written $obdd_\chi$, is defined inductively on χ as follows:

- $obdd_\top$ is the OBDD consisting of a single node labeled T;
- $obdd_\bot$ is the OBDD consisting of a single node labeled F;
- $obdd_p$ is the following OBDD:

Input: MOBDDs $mobdd$ and $mobdd'$, an ordering $<$ on all atoms of $mobdd$ and $mobdd'$ compatible with their associated orderings. A truth–functional connective $\star$.

Output: A MOBDD denoted $\texttt{Apply}(mobdd, mobdd', \star, <)$.

1. Take the OBDDs $obdd$ and $obdd'$ generated by the entry points of $mobdd$ and $mobdd'$. Let p and p' be the labels of the entry points of $obdd$ and $obdd'$ respectively.

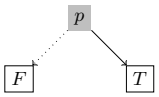
- If $obdd$ and $obdd'$ are both leaves, define the leaf $mobdd_0$ labeled by $p \star p'$.
- If $p = p'$, define the OBDD $mobdd_0$ whose entry point is labeled by p, whose left sub-OBDD is $\texttt{Apply}(obdd_l, obdd'_l, \star, <)$ and whose right sub-OBDD is $\texttt{Apply}(obdd_r, obdd'_r, \star, <)$.
- if $p < p'$, define the OBDD $mobdd_0$ whose entry point is labeled by p, whose left sub-OBDD is $\texttt{Apply}(obdd_l, obdd', \star, <)$ and whose right sub-OBDD is $\texttt{Apply}(obdd_r, obdd', \star, <)$.

  Otherwise, define the OBDD $mobdd_0$ whose entry point is p', whose left sub-OBDD is $\texttt{Apply}(obdd, obdd'_l, \star, <)$ and whose right sub-OBDD is $\texttt{Apply}(obdd, obdd'_r, \star, <)$.

where $obdd_l, obdd'_l$ (resp. $obdd_r, obdd'_r$) are the left (resp. right) sub-BDDs of $obdd$ and $obdd'$.

2. Return $obdd_0$ [with the disjoint union of the OBDDs generated by the other roots of $mobdd$ and $mobdd'$].

Figure 6: Schematic algorithm `Apply` for OBDD [and MOBDD].



- if $\chi = \varphi \star \psi$, then $obdd_\chi \triangleq \texttt{Apply}(obdd_\varphi, obdd_\psi, \star, <)$.
- if $\chi = \neg\varphi$, then $obdd_\chi$ is $obdd_\varphi$ where the labels T or F of the leafs have been switched.

Example 3. Let us consider the formula $\varphi \triangleq p \rightarrow ((q \wedge \neg r) \vee \neg s)$. The syntactic decomposition tree of formula φ is represented at the top of Figure 7. From this decomposition tree, we can apply the induction process of Definition 8 to obtain

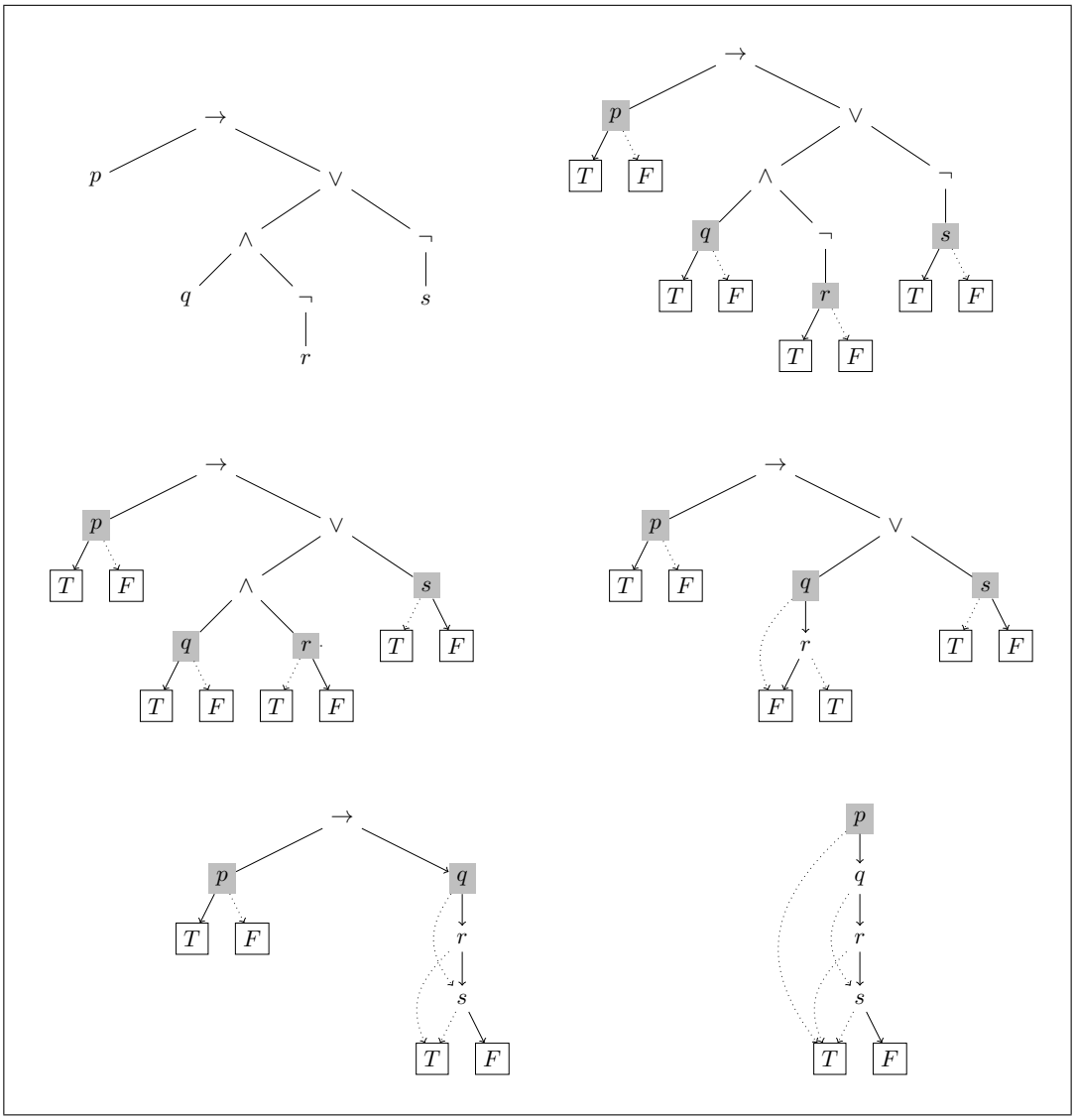


Figure 7: Successive applications of `Apply` with the ordering $p < q < r < s$: from the syntactic tree of $\varphi \triangleq p \to ((q \wedge \neg r) \vee \neg s)$ (*top left*) to $obdd_\varphi$ (*bottom right*).

Input: A BDD *obdd* for a formula φ; an interpretation $\mathcal{I} \in \mathcal{C}^p$ (partial or total).

Output: A BDD denoted `Restrict`$(\mathcal{I}, obdd)$.

Perform a recursive traversal of the BDD:

- If the root of *obdd* is a leaf, return the leaf.
- If the root of *obdd* is labeled p and $\mathcal{I}(p)$ is defined, return the sub-BDD reached by its true edge if $\mathcal{I}(p) = T$ and the sub-BDD reached by its false edge if $\mathcal{I}(p) = F$.
- Otherwise (the root of *obdd* is labeled p and $\mathcal{I}(p)$ is *not* defined), apply the algorithm to the left and right sub-BDD, and return the BDD whose root is p and whose left and right sub-BDD are those returned by the recursive calls.

Figure 8: Schematic algorithm `Restrict`.

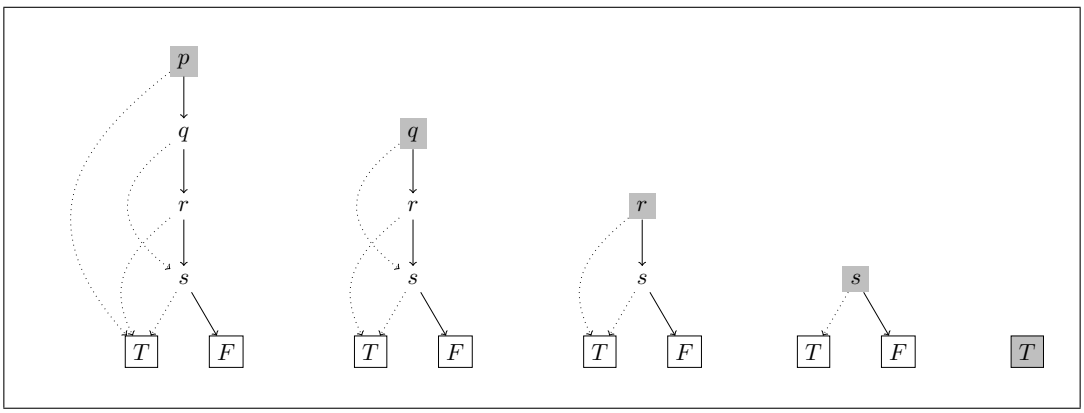


Figure 9: Successive applications of `Restrict` to $obdd_\varphi$ of Figure 7 with the interpretations $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}_3$ and $\mathcal{I}_4$.

the OBDD $obdd_\varphi$. This process is represented in Figure 7, it consists in applying iteratively algorithm `Apply` of Figure 6.

Definition 9 (Operation `Restrict`)**.** The operation `Restrict` on BDD and interpretations is defined in Figure 8.

Example 4. Figure 9 shows the successive application of `Restrict` on the OBDD $obdd_\varphi$ of Figure 7, with the interpretations $\mathcal{I}_1(p) = T$ and $\mathcal{I}_1(t)$ undefined for all

$t \neq p$, $\mathcal{I}_2(q) = T$ and $\mathcal{I}_2(t)$ undefined for all $t \neq q$, $\mathcal{I}_3(r) = T$ and $\mathcal{I}_3(t)$ undefined for all $t \neq r$, $\mathcal{I}_4(s) = F$ and $\mathcal{I}_4(t)$ undefined for all $t \neq s$.

The key properties of the algorithms `Restrict` and `Reduce` are highlighted by the following results.

Proposition 1. *Let $obdd_\varphi$ be an OBDD associated to a formula $\varphi \in \mathcal{L}$ (and an ordering) and let $\mathcal{I} \in \mathcal{C}^p$ be an interpretation. Then, `Restrict`$(\mathcal{I}, obdd_\varphi)$ returns an OBDD associated to a formula ψ such that $\varphi \leftrightarrow \psi$ is valid on the set of interpretations extending $\mathcal{I}$, $Ext(\mathcal{I})$.*

Theorem 2. *The algorithm `Reduce` constructs an OBDD if the original BDD is ordered. For a given ordering of atoms, the reduced OBDDs for logically equivalent formulas are structurally identical.*

All operations on OBDD, `Apply`, `Reduce`, and `Restrict`, have a polynomial algorithmic complexity with the size of the OBDD they operate on. The size of the result of `Reduce` strongly depends on the atom ordering. It is exponential in the worst case. Some formulas even have an exponential size OBDD for all orderings. However, OBDD have shown to be of great pratical value, and have become a standard solution for dealing with industrial size propositional formulas, *e.g.* in the areas of digital system design, verification and testing [10].

3 The examination operator

Sometimes during a trial, the judge must necessarily examine or raise a specific issue. For example, the complainant can attack the legitimity of a trade union presenting a candidate to the professionnal elections of a firm. The complainant could argue that this association of employees cannot really be qualified as a syndicate because it is not senior enough. In that case, even if the trade union turns out to be senior enough (older than 2 years), the judge *must* nevertheless examine *all* the other criteria (different from seniority) that make the association of employees qualify as a trade union (even if she/he does not decide to raise the issue of the other criteria during the trial).

Definition 10 (Language $\mathcal{L}_M$)**.** The language $\mathcal{L}_M$ is the set that contains $\mathbb{P} \cup \{\top, \bot\}$ and such that

- if $\varphi, \psi \in \mathcal{L}_M$, then $!\varphi, \neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi) \in \mathcal{L}_M$;
- $\mathcal{L}_M$ contains no more formulas.

We use the same abbreviations and notations as in Definition 1.

The intuitive reading of the formula $!\varphi$ is as follows: "φ is examined and it holds". For example, the formula $q \wedge !p$ holds if, and only if, p is examined and p and q both hold.

We must provide a semantics to our extended language $\mathcal{L}_M$, in particular to the examination operator $!\varphi$. The Algorithm `Apply` provides already a semantics to every formula of the propositional language in terms of OBDD. Indeed, it suffices to apply it inductively to every subformula of a given formula φ (from the atoms up to the formula φ) to obtain an OBDD that captures the meaning of φ. Thus, we extend this algorithm to include the examination operator as well. With this aim in view, we introduce an extended form of BDD with several entry points. This extension is called a *Multi-BDD* and, contrary to BDD, it can have several roots.

Definition 11 (Multi-BDD, Multi-OBBB)**.** A *Multi-BDD* (MBDD for short) is a BDD with possibly multiple roots $r_0, \ldots, r_k$. The root r_0 is distinguished and called the *entry point* of the MBDD. A *Multi-OBDD* (MOBDD for short) is a MBDD such that the orderings of atoms $<^0, \ldots, , <^k$ defined by the branches stemming from the roots $r_0, \ldots, r_k$ are compatible. A MBDD or MOBDD is *elementary* when it consists only of leaves.

Example 5. Figure 13 shows a MOBDD. The construction of this MOBDD is detailed in the next section.

Definition 12 (MOBDD associated to a formula and an ordering)**.** Let $\chi \in \mathcal{L}_M$ and let $<$ be an ordering on the set of atoms occuring in χ. The MOBDD associated to χ (and $<$), written $mobdd_\chi$, is defined inductively on χ as follows.

- $mobdd_\top \triangleq obdd_\top$, $mobdd_\bot \triangleq obdd_\bot$, $mobdd_p \triangleq obdd_p$ of Definition 8.
- if $\chi = (\varphi \star \psi)$, then
  - if $\varphi \neq !\varphi'$ and $\psi \neq !\psi'$ for any $\varphi', \psi' \in \mathcal{L}_M$, then $mobdd_\chi \triangleq \texttt{Apply}(mobdd_\varphi, mobdd_\psi, \star, <)$ (as defined in Figure 6);
  - if $\varphi = !\varphi'$ and $\psi \neq !\psi'$ for some $\varphi' \in \mathcal{L}_M$ and any $\psi' \in \mathcal{L}_M$ (or $\varphi \neq !\varphi'$ and $\psi = !\psi'$ for some $\psi' \in \mathcal{L}_M$ and any $\varphi' \in \mathcal{L}_M$), then $mobdd_\chi \triangleq mobdd_{\varphi'} \sqcup \texttt{Apply}(mobdd_{\varphi'}, mobdd_\psi, \star, <)$ (resp. $mobdd_\chi \triangleq mobdd_{\varphi'} \sqcup \texttt{Apply}(mobdd_\varphi, mobdd_{\psi'}, \star, <)$;
  - if $\varphi = !\varphi'$ and $\psi = !\psi'$ (for some $\varphi', \psi' \in \mathcal{L}_M$), then $mobdd_\chi \triangleq mobdd_{\varphi'} \sqcup mobdd_{\psi'} \sqcup \texttt{Apply}(mobdd_{\varphi'}, mobdd_{\psi'}, \star, <)$.

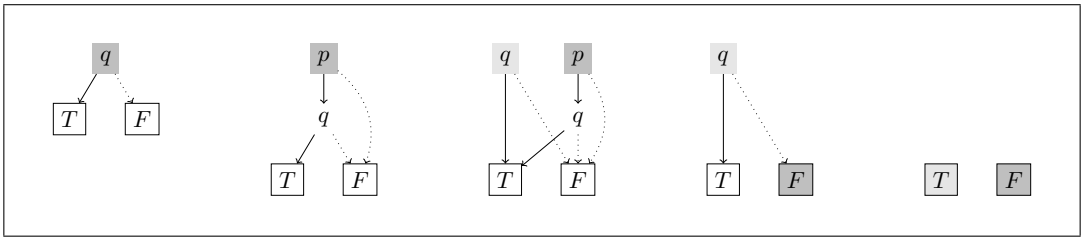


Figure 10: $obdd_q$ (*first*), $obdd_{p\wedge q}$ (*second*), $mobdd_{(p\wedge !q,<)}$ reduced (*third*), $\texttt{Restrict}\left(\mathcal{I}, mobdd_{(p\wedge !q,<)}\right)$ reduced, with $\mathcal{I}(p) = F$ and $\mathcal{I}(q)$ undefined (*fourth*), $\texttt{Restrict}\left(\mathcal{I}, mobdd_{(p\wedge !q,<)}\right)$ with $\mathcal{I}(p) = F$ and $\mathcal{I}(q) = T$ (*fifth*) with ordering $p < q$.

- if $\chi = \neg\varphi$, then $mobdd_\chi$ is $mobdd_\varphi$ where the labels T or F of the leafs have been switched.

It is important to notice that in order to reduce a MOBDD to an elementary MOBDD, the magistrate must evaluate *every* formula associated to a root of the MOBDD. The proposition below shows that our definition of MOBDD does capture that requirement.

Proposition 2. *Let $\varphi \in \mathcal{L}_M$ with subformula $!\psi$ and let $\mathcal{I} \in \mathcal{C}^p$ be a partial interpretation. If $\mathcal{I}(\psi) \notin \{T, F\}$, then $\texttt{Reduce}(\texttt{Restrict}(\mathcal{I}, mobdd_\varphi))$ is not elementary.*

Example 6. In Figure 10, we represent the MOBDD associated to the formula $p\wedge!q$ and ordering $p < q$ (disjoint union of first and second OBDD, reduced in the third graph). Roots are in gray and the entry points are darker. This formula is true if, and only if, p and q are both true and q is examined. Hence, if p is given the value F (by the judge), then, even if the truth value of the formula $p\wedge!q$ is determined, the MOBDD is still not elementary. Indeed, we must *examine* the 'question' q and give a truth value to q. Then, once the judge has examined question 'q', we reach the fifth MOBDD, which is elementary.

4 Case study: French trade unions

Our case study deals with French professional election in a firm [38]. Among the problems to be decided upon, one is to determine whether an association of employees is really qualified as a trade union so that this association can propose employees to come forward as candidates at the professional elections of the firm. Law introduces four criteria that have to be fulfilled so that an association can indeed be

qualified as a trade union. These four criteria have to hold altogether, and during a trial, the judge must necessarily examine all of them. They are the following:

1. the association of employees should respect the 'Republican values';
2. the association of employees should be 'Independent' (from the directorate of the firm for example);
3. the association of employees should be 'Senior' enough (minimum 2 years of existence);
4. the association of employees should be within the appropriate 'Geographical and professional range'.

Hence, we introduce the formula R (resp. I, S and G) which stands respectively for "the criteria of the *Republican values* (resp. *Independence*, *Seniority*, *Geographical and professional range*) is fulfilled". The four criteria must hold for an association of employees to be legally qualified as trade union in a firm, and they must all be examined by the judge, even if they are not contested. Therefore, the following formula must be true: $(!R \wedge (!I \wedge (!G \wedge !S)))$.

4.1 The criterias of 'republican values' and 'independence'

To determine whether the criteria of 'Republican values' holds, the judge has to answer a number of questions. To formulate these questions, we introduce the following set of atoms:

$$\mathbb{P}_R \triangleq \{Lit_R, OldJug_R, NewElt_R, Proof_{\neg R}\}$$

These atoms stand for the following propositions:

- Lit_R: "The plaintiff contests the criteria of 'Republican values' ";
- $OldJug_R$: "An old judgement dealing with the criteria of 'Republican values' already established that the association of employees fulfills that criteria";
- $NewElt_R$: "New elements have been brought to the fore that oblige to reconsider the old judgement";
- $Proof_{\neg R}$: "The plaintiff presents the proof that the criteria of 'Republican values' is not fulfilled".

Then, we can give the formula in the language $\mathcal{L}_M$ that determine in which case the criteria of 'Republican values' holds. It is the following:

$$R \triangleq Lit_R \rightarrow ((OldJug_R \wedge \neg NewElt_R) \vee \neg Proof_{\neg R})$$

The above formula reads as follows: "if the plaintiff contests that the criteria of 'Republican values' is satisfied, then either there is an old judgement which already established that this criteria was fulfilled and no new elements have been brought to the fore which oblige to reconsider this old judgement, or there is no old judgement and the plaintiff has not been able to prove that the criteria is not fulfilled". However, in the procedure that the judge must follow during a trial, he must first determine whether or not there was an old judgment (that already established that this criteria was fulfilled) before asking the plaintiff to provide a proof that the criteria is not fulfilled. This procedural reasoning is captured by our OBDDs. In the left OBDD of Figure 11, the judge first has to check that there was an old judgement establishing the criteria. In the right OBDD of Figure 11, he first has to ask the plaintiff to provide a proof that the criteria is not fulfilled (without wondering whether an old judgement was already established or not).

Dealing with the criteria of 'Independence' is completely similar. Hence, we introduce the following set of atoms:

$$\mathbb{P}_I \triangleq \{Lit_I, OldJug_I, NewElt_I, Proof_{\neg I}\}$$

Their interpretation is the same as for the criteria of Republican values, except that the term "Republican values" has to be replaced by "Independence" everywhere. So, the formula I of the language $\mathcal{L}_M$ which determines in which case I holds is the following:

$$I \triangleq Lit_I \rightarrow ((OldJug_I \wedge \neg NewElt_I) \vee \neg Proof_{\neg I})$$

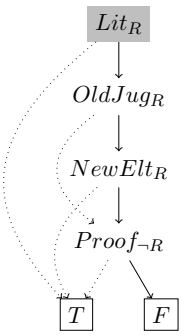
Its intuitive interpretation is the same as for the previous criteria.

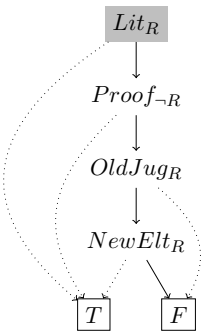
4.2 The criteria of 'geographical and professional range'

For the criteria of 'Geographical and professional range', we introduce the following set of atoms:

$$\mathbb{P}_G \triangleq \{Lit_G, Decide_G, Proof_{\neg G}\}$$

These atoms stand for the following propositions:





$\mathcal{I}(Lit_R)$	$\mathcal{I}(OldJug_R)$	$\mathcal{I}(NewElt_R)$	$\mathcal{I}(Proof_{\neg R})$	$\mathcal{I}(R)$
T	T	T	T	F
T	T	T	F	T
T	T	F	T	T
T	T	F	F	T
T	F	T	T	F
T	F	T	F	T
T	F	F	T	F
T	F	F	F	T
F	T	T	T	T
F	T	T	F	T
F	T	F	T	T
F	T	F	F	T
F	F	T	T	T
F	F	T	F	T
F	F	F	T	T
F	F	F	F	T

Figure 11: Three logically equivalent representations. OBDD associated to $R = Lit_R \rightarrow ((OldJug_R \wedge \neg NewElt_R) \vee \neg Proof_{\neg R})$ and $Lit_R < OldJug_R < NewElt_R < Proof_{\neg R}$ (*top left*), $Lit_R < Proof_{\neg R} < OldJug_R < NewElt_R$ (*top right*). Truth table of R (*bottom*).

- Lit_G: "The plaintiff contests the criteria of 'Geographical and professional range' ";
- $Decide_G$: "The judge decides to examine the criteria of 'Geographical and professional range' ";
- $Proof_{\neg G}$: "The plaintiff presents the proof that the criteria of 'Geographical

and professional range' is not fulfilled".

The formula G of the language $\mathcal{L}_M$ determines in which case the criteria of 'Geographical and professional range' holds:

$$G \triangleq (Lit_G \vee Decide_G) \rightarrow \neg Proof_{\neg G}$$

The formula G reads as follows: "if the plaintiff contests that the criteria of 'Geographical and professional range' is fulfilled or if the judge decides to consider this criteria, then the plaintiff presents the proof that the criteria is not fulfilled". The OBDD associated to the formula G is depicted in Figure 12, it is the third OBDD from the left. In that last example, even if the criteria was not contested by the plaintiff, the judge can decide to solicit or not the plaintiff on that issue.

Logically equivalent representations. In Figure 11, we provide three equivalent and alternative representations of the semantics of the formula R: the truth table of R and two OBDDs associated to R that only differ on their associated orderings. The construction of the top left OBDD using the algorithm `Apply` is given in Figure 7 (with the appropriate atom substitutions). During a trial, the judge answers the questions corresponding to the nodes of the OBDD in the order specified by the ordering of the OBDD. However, this order could be changed automatically thanks to the algorithms dedicated to OBDDs, if needed. He could also 'navigate' in the binary decision diagram to explore it without having to answer any question node. Each time the judge answers a question, the OBDD is updated to a simpler OBDD. This update is performed by the `Restrict` algorithm.

Example 7. A series of questions answering and updates is given in Figure 9 (with the appropriate atoms substitutions). First, the judge answers 'yes' to question p (alias Lit_R) because the plaintiff contests the criteria. The software tool then applies the `Restrict` algorithm on the first OBDD of Figure 9 with the interpretation $\mathcal{I}_1$, yielding the second OBDD of Figure 9. Then, the judge answers 'yes' again to the question q (alias $OldJug_R$) because an old judgement dealing with the criteria has indeed already established its validity. The software tool then applies the `Restrict` algorithm to the second underlying OBDD of Figure 9 with $\mathcal{I}_2$, yielding the third OBDD of Figure 9. Then, he answers 'yes' to r (alias $NewElt_R$) because new elements have been brought to the fore that oblige to reconsider the old judgement, yielding the fourth OBDD of Figure 9 by application of `Restrict` with $\mathcal{I}_3$. Finally, he answers 'no' to s (alias $Proof_{\neg R}$) because these new elements do not invalidate the old judgement establishing the validity of the criteria. The software tool finally reaches the last elementary OBDD of Figure 9 by application of `Restrict` with $\mathcal{I}_4$, stating that the criteria of 'republican values' is fulfilled in that particular litigation.

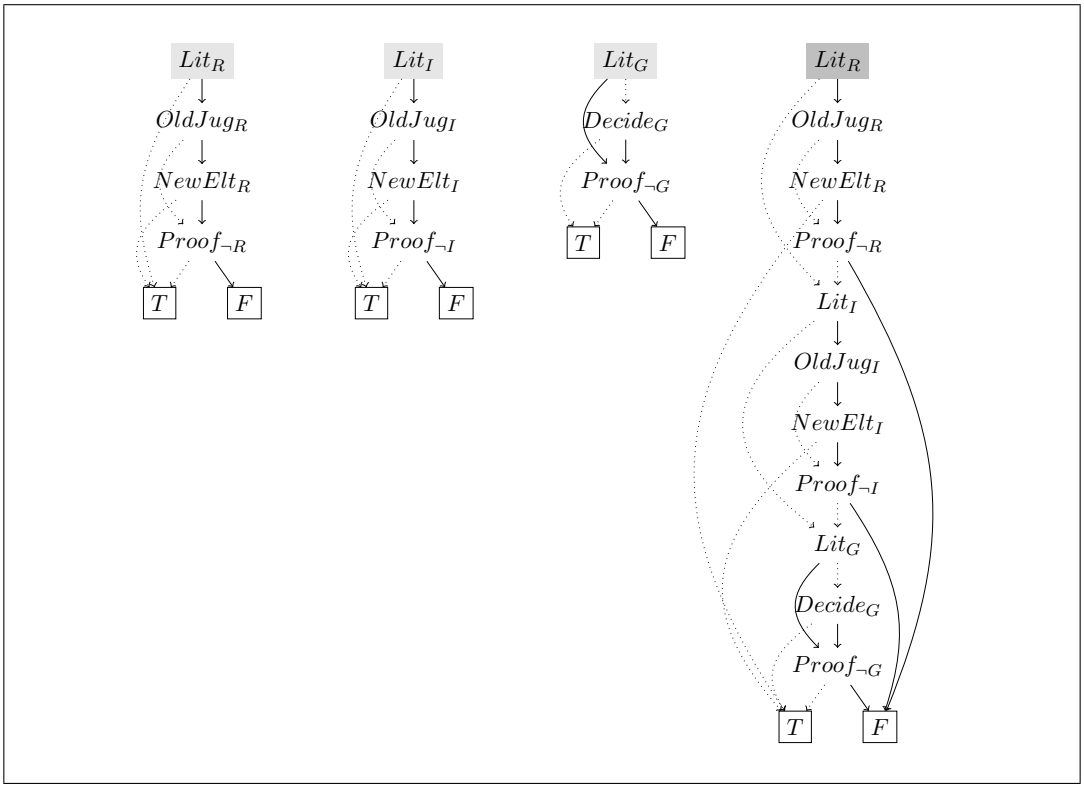


Figure 12: MOBDD of $(!R \wedge (!I \wedge !G))$ with ordering $Lit_R < OldJug_R < NewElt_R < Proof_{\neg R} < Lit_I < OldJug_I < NewElt_I < Proof_{\neg I} < Lit_G < OldJug_G < NewElt_G < Proof_{\neg G}$. The four OBDD are associated to R, I, G and $(R \wedge (I \wedge G))$, the entry point is in dark gray.

4.3 A MOBDD for qualifying as a trade union

We have not considered the criteria of 'Seniority' until now and we will not deal with it in order to ease the presentation and because it is more complex to represent than the other criteria, in the sense that many more atoms (questions) are needed to deal with it. The MOBDD of $(!R \wedge (!I \wedge !G))$ with ordering $Lit_R < OldJug_R < NewElt_R < Proof_{\neg R} < Lit_I < OldJug_I < NewElt_I < Proof_{\neg I} < Lit_G < OldJug_G < NewElt_G < Proof_{\neg G}$ is represented in Figure 12. It is simply the disjoint union of the three OBDD associated to R, I, G and $(R \wedge (I \wedge G))$. Then, it can be reduced equivalently thanks to the Algorithm `Reduce` of Figure 4 to the MOBDD of Figure 13. Note that in this MOBDD, all the leaf nodes have been merged into two nodes ($obdd_\top$ and $obdd_\bot$).

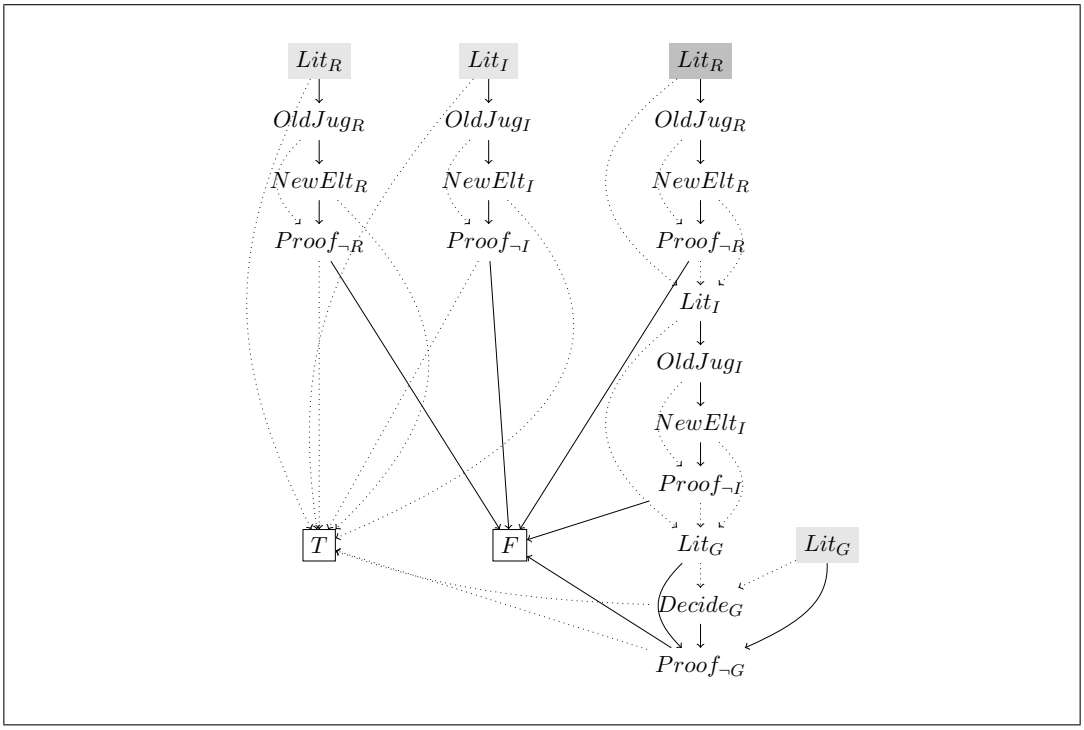


Figure 13: MOBDD of $(!R \wedge (!I \wedge !G))$ of Figure 12 reduced.

5 Applications of BDD algorithms to legal reasoning

Because our solution is based on BDD, we inherit from the vast amount of work for BDD a number of algorithms and software applications that can play an important role in legal reasoning. Even if they were not initially intended to be used in the legal domain, these algorithms and software applications turn out to be really relevant for solving specific problems or answer specific queries of the judge. We list below some of these algorithms (some of them have already been considered above) and show how they can be used by a judge during, before or after a trial. We start with the algorithms of this article:

- Algorithm `Restrict`. This algorithm can be used when the judge answer questions: each question answered corresponds in fact to an application of the algorithm `Restrict`. After each answer, the MBDD is instanciated and the node corresponding to the question disappears (see Example 7).

- Algorithm `Reduce`. This algorithm can be used to determine whether two kinds

of legal reasoning represented by two different MBDD are in fact equivalent: in case the MBDD returned by this algorithm is the same in both cases, then they are indeed equivalent (see Theorem 2).

- Algorithm `Apply`. This algorithm can be used to construct a MOBDD corresponding to a formula expressed in our language $\mathcal{L}_M$. It can also be used when we want to combine two kinds of legal reasoning that deal with different but complementary issues that have already been represented by two MOBDD.

Other algorithms based on BDD dealing with quantification over propositional atoms can be used to solve the following tasks:

- Determine whether the answer to a specific question will allow the judge to conclude about a litigation or a specific subproblem without having to examine all the other questions exhaustively.

- Determine whether a question is redundant and can thus be removed from the MBDD.

These algorithms are only a few among the large amount of algorithms for BDD which are available. Many other algorithms can be used or designed or derived to solve specific legal reasoning tasks.

6 Our prototype

In this section, we introduce the prototype that was developed during our project. Our prototype does not use any BDD software library: we realized that the graphs elaborated in collaboration with the jurists were in fact BDD at a rather well-advanced stage of the project. The global architecture of the prototype is given in Figure 14. The latest version of our prototype is available at the following address:

`http://cassation.gforge.inria.fr/prototype-2015-06-08`

To try it out, please contact one of the authors to obtain the access codes.

In Section 6.1, we describe the front-end graphical user interface. The development process was iterative (inspired by the agile method) and Section 6.2 explains how the graphs were designed interactively by the jurists and computer scientists for creating the input graph. In Section 6.3, we introduce the architecture of the graph generation from `.doc` and `.dia` files using Microsoft$^{\text{TM}}$ Word and the software Dia. In Section 6.4, we describe the architecture of the front-end that takes the graph as input.

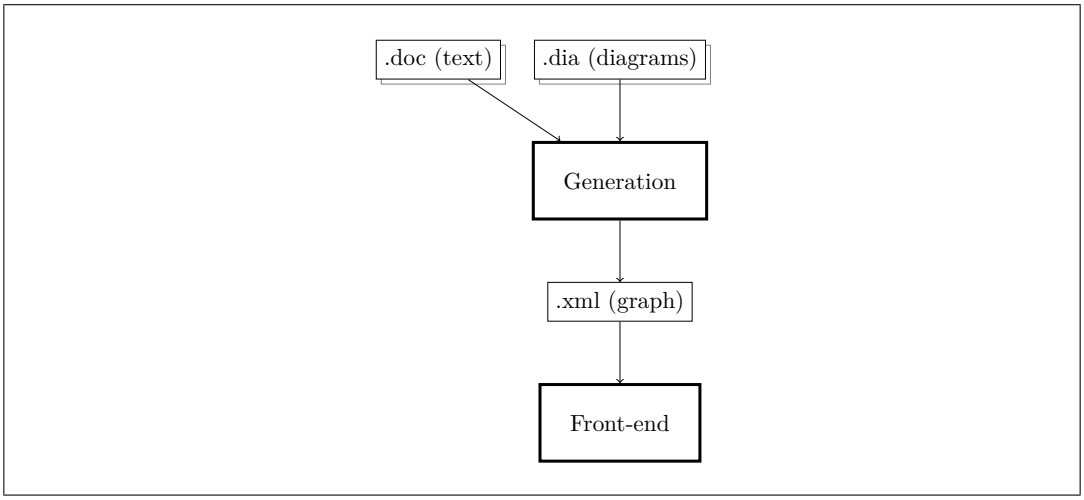


Figure 14: Global architecture

6.1 Graphical user interface

The graphical user interface is represented in Figure 1 and the front-end is split up into two parts. The left part proposes questions to be answered and the documentation for helping the judge to make decisions. While the judge answers questions, the right part shows the produced judgment text.

6.2 Iterative design method

During the project, computer scientists and jurists needed to find out a common 'language'. Jurists first resorted to graphical representations of their reasonings under the form of binary decision trees, which are more natural than the formulas and truth tables of logic. At some point, to ease communication, we decided to adopt a common code and use a software editor called Dia (`http://dia-installer.de/`). The jurists had quite some flexibility and autonomy. For instance, they could decide to adopt new symbols for new concepts whenever they felt that it was needed. It was very instructive, both for computer scientists and jurists, to discuss and exchange ideas based on these diagrams during several meetings. We agreed on a set of graphical conventions. The textual documentation was already created by jurists with informal structural conventions. It consisted in a collection of `.doc` files and we decided to keep this format.

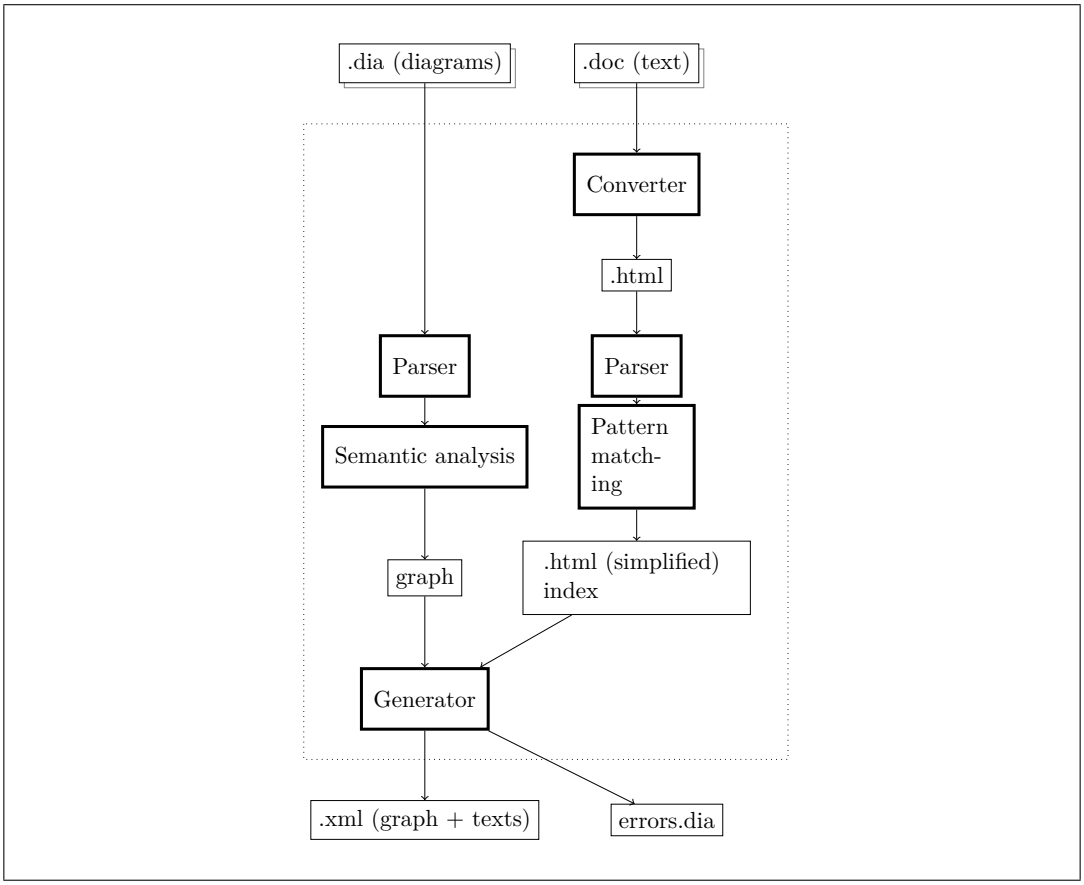


Figure 15: Graph generation architecture.

6.3 Graph generation architecture

Figure 15 represents the graph generation architecture. The input is divided in two parts: diagrams in `.dia` and the documentation in `.doc` files. On the one hand, BDD are represented by diagrams saved in `.dia` files. We used some graphical conventions to encode the various elements (questions, answers, connectives). The diagrams are parsed and a semantic analysis generates a graph from diagrams by using the graphical conventions. On the other hand, `.doc` files are converted in `.html` files, which are then parsed. We identify the different paragraphs in `.html` files and we produce a single simplified `.html` file and an index identifies the sections and the judgment paragraphs.

The last step generates a `.xml` file which merges the graph and the textual

documentation (sections and judgment paragraphs). Errors are reported graphically in a `.dia` file.

6.4 Front-end architecture

The front-end is implemented in Javascript and is fully executed in the web browser. The front-end is based on a model-view-controller architecture. The model is dynamically generated from the `.xml` file given as input. The generated `.xml` file contains all the data for displaying questions and judgment paragraphs in the front-end. We can export the produced judgment output as a text file.

7 Conclusion

One could argue that our proposed solution is not really suitable because it is merely based on propositional logic and does not integrate non-classical reasoning such as deontic, causal or defeasible reasoning, which has often been claimed to be more appropriate to deal with legal reasoning (see the various references in Section 1.1 and [13]).[5] As such, our work is only a first step and it is quite possible to extend it to other kinds of reasoning hardly amenable to propositional reasoning. In fact, even if we have not addressed in this article standard problems in AI and law such as those related to 'contrary to duties' or exceptions, we did encounter such kinds of exceptional reasoning in the course of our project. As it turns out, exceptions could also be dealt with BDD and we actually introduced with that aim in view in the course of our project an operator called "sinon" ("otherwise" in English) with a semantics based on BDD. Our solution for handling exceptions based on BDD turned out to be also very appealing to jurists.

This research was carried out from the outset hand in hand by both jurists and computer scientists, with regular and numerous communications between both parties. The graphical representation that we made up and shared for our common work and formalization turned out in the end to correspond to BDD. This type of representation was more along the lines of the actual practice of the jurists. In particular, the procedural and temporal aspect of BDD was in fact very close to their actual practice as lawyers or jurists: in a BDD, one has to answer one after the other questions corresponding to the nodes of the BDD. This procedural and

[5] Quite different approaches based on neural networks and machine learning have been proposed to justify *a posteriori* the reasoning of the judge, and also predict it, such as for example the work of Borges and Bourcier [19]. However, these other kinds of models could not really be used to solve the problems that concerned us here because they do not really provide means to help judges to form their judgements: the crucial reasoning part is absent from these models.

temporal aspect of the legal practice cannot be captured by the usual syntactic representation of propositional formulas. We made an experiment with law students of the Ecole Normale Supérieure of Rennes to determine whether they prefer to write the kind of meta-regulations that we use in the software tool with formulas of (propositional) logic or with a graph-based representation like BDD. We designed a kind of experimental protocol to figure this out. Even if the results were hard to interpret because they had no prior teaching in logic or graph theory, it turns out that they had somehow more facility to use the graph-based representation than the formula-based representation.

Propositional logic is not intended to model the current state of legal texts and regulations. It only serves here as a theoretical basis for the *rewriting* of most of legal texts and regulations together with their dual representation in terms of BDD.[6] In fact, this graphical representation in terms of BDD could also be extended to more complex kinds of legal reasoning, as mentionned above. Hence, criticisms regarding the adequacy of propositional logic for legal reasoning do not really apply to our work, especially in this specific context of the development of a judgement editor. This rewriting of legal texts and regulations in logical terms can be viewed as a new type of codification (see Section 1.2). This new codification would then provide theoretical basis for the development of software tools that could be used by jurists and lawyers, and probably change their actual practice of the law. If our proposal is adopted, the current legal texts and regulations would have to be all rewritten and adapted in order to fit the format based on BDD propounded in this article, as we did during our project with the case study of the "French trade unions" (see Section 4). The rewriting and adaptation phase could start with small fragments of the law and it could then be expanded step by step to all parts of the law.

From a theoretical point of view, we believe that our solution is the most promising and realistic approach to answer the needs and problems summarized in Section 1.2. Indeed, it is based on methods and techniques of logic that are very well understood, worked out and applied and therefore provides a rigorous and solid foundation to subsequent technological developments. Our approach has the advantage to provide a strong control over our representation of the legal reasoning and over the changes that we may want to make to this reasoning. Moreover, BDD are very well–studied and their associated algorithms are able to scale–up to a large number

[6] In [6, 8, 7], we propounded another rewriting of legal texts and regulations in logical terms in order to deal with problems of privacy. The proposed reformulation was different from a logical point of view, somehow more specific, because it was meant for other legal purposes in a particular context, namely to check that the privacy policies declared by a company or organization are compliant with respect to the privacy regulations of a given legislation and to check that the company or organization does enforce its declared privacy policy over the internet.

of nodes. Thus, using BDD is clearly a realistic solution to deal with the complexity of the law and the large amount of texts and jurisprudences. Finally, our approach is very flexible and can take into account the dynamism and unpredictable changes of the law. Indeed, because it is based on propositional logic, the various changes in the law, such as promulgation, abrogation and annulment can be modeled naturally as update operations in propositional logic. Historically, the well-known AGM theory of belief change [3] was propounded by three researchers whose one of them, Alchourrón, was a jurist: AGM theory from his point of view was supposed to model and deal with changes in the law, viewed as a theory of propositional logic, such as promulgation and abrogation in an abstract way.[7] As it turns out, dealing with dynamism and change has been at the core of most of the recent developments in logic and artificial intelligence in the last decades and many extensions of propositional logic with dynamic operators have been introduced (see for instance [23, 40, 41, 9]). Hence, the dynamic character of the law could be dealt within the software by importing, adapting and implementing the various methods and techniques which have been developed in logic for dealing with dynamism and change.

Our case study based on the "French trade unions" (Section 4) shows that our approach is feasible and can be extended to any kind of regulations, even if a tremendous amount of work would need to be carried out by the jurists (in collaboration with computer scientists and logicians) to rewrite and adapt all the existing texts and regulations in order to define corresponding BDD. Our case study has been implemented in a prototype in the course of our project (see Section 6). This prototype was tried out by four judges of 'tribunaux d'instance' in France accustomed to our case study. They were all rather impressed and satisfied with the prototype tool. This said, the software tool introduced here is only the first part of a larger project since this software tool would only *use* the BDD representing the legal reasoning underlying specific litigations. But these BDD would first need to be defined and created by a legal expert or a legislator. The second tool that complements the software described in this article still has to be specified precisely. Its role would be to create and edit these BDD that capture any other case studies. In particular, this second software tool should provide mechanisms for modifying the graphs that underly the BDD. From a theoretical point of view, graph modifications and change is also an area of research that is currently very active in logic [5, 9, 24, 4, 12]. These theoretical works could provide algorithms and associated software tools for checking and verifying that a particular change or modification of the BDD has indeed been made and that these changes do correspond to the idea and the intention of the user/legislator who triggered them.

[7] Governatori & Al. [26, 25] attempt to model abrogation and annulment more realistically.

Finally, if such kinds of softwares would ever be developed and used by judges, this would entail as a prerequisite that the jurists be trained and taught some rudiments of logic, especially the law experts who would have to rewrite and adapt the legal texts and regulations to create and edit the corresponding BDD. This would also call for the enactement of regulations to determine in which kinds of legal context and litigations these softwares can and should be used. Indeed, the novelty of such softwares and their impact on society would raise a number of ethical issues that would need to be harnessed by the law. This said, we want to stress that this work will not replace by any means judges by 'machines', nor will suppress the responsibility that judges endorse when they make decisions. In this article, we only propose some theoretical foundations that could lead to the development of a software tool to help judges to make fair, well-informed and maybe sometimes better decisions. This software tool would also provide and recall judges some well–structured information about the relevant legal texts and jurisprudence that they should not omit, together with some crucial information about the legal procedure that should be followed in order to deal with a specific litigation. If such a software tool were available some day to judges, they should in any case remain fully responsible of their decisions and they should not have the possibility to discharge the responsibility of their decisions to that software tool.

Acknowledgments. The work reported in this article was carried out in the context of a collaborative project with the Cour de cassation. The coordinator of this project was Guillaume Aucher.[8] We thank Anthony Baire, Annie Foret, Jean-Baptiste Lenhof and François Schwarzentruber for their participation. We thank François Schwarzentruber for developing the very first prototype and Anthony Baire for developing the subsequent prototypes. We thank Marie-Pierre Lanoue, Daniel Tardif, Eloi Buat-Menard, Laurence Pecaut-Rivolier, Llio Humphreys, Guido Boella as well as Jean-Paul Jean, Damien Pons and Ronan Guerlot for their advices and participation and for contributing to make this project happen. We thank Aude Bubbe, Laurence Pecaut-Rivolier, Aurélie Police and Françoise Simond for trying out our prototype. We thank Olivier Ridoux for carefully reading the article and for proposing to use multi–BDD instead of three–valued OBDD.

References

[1] Ajani, G., Boella, G., Caro, L. D., Robaldo, L., Humphreys, L., Praduroux,

[8] Guillaume Aucher was interviewed at the outset of the project by Jean-Michel Prima: `http://emergences.inria.fr/lettres2013/newsletter-n28/L28-OUTILDECISION`.

S., Rossi, P., and Violato, A. The European taxonomy syllabus: A multi-lingual, multi-level ontology framework to untangle the web of european legal terminologyuropean taxonomy syllabus: A multi-lingual, multi-level ontology framework to untangle the web of european legal terminology. *Applied Ontology 11*, 4 (2016), 325–375.

[2] Alchourrón, C. E., and Bulygin, E. *Normative systems*. Springer-Verlag, Wien, New York, 1971.

[3] Alchourrón, C. E., Gärdenfors, P., and Makinson, D. On the logic of theory change: Partial meet contraction and revision functions. *J. Symb. Log. 50*, 2 (1985), 510–530.

[4] Areces, C., Fervari, R., and Hoffmann, G. Relation-changing modal operators. *Logic Journal of the IGPL 23*, 4 (2015), 601–627.

[5] Aucher, G., Balbiani, P., Cerro, L. F. D., and Herzig, A. Global and local graph modifiers. In *Methods for Modalities 5 (M4M-5)* (Cachan, France, 2007), ENTCS, Elsevier.

[6] Aucher, G., Barreau-Saliou, C., Boella, G., Blandin-Obernesser, A., Gambs, S., Piolle, G., and Van Der Torre, L. The Coprelobri project : the logical approach to privacy. In *2e Atelier Protection de la Vie Privée (APVP 2011)* (Sorèze, France, June 2011).

[7] Aucher, G., Boella, G., and van der Torre, L. Privacy policies with modal logic: the dynamic turn. In *Deontic Logic in Computer Science (DEON 2010)* (2010), G. Governatori and G. Sartor, Eds.

[8] Aucher, G., Boella, G., and van der Torre, L. A dynamic logic for privacy compliance. *Journal of artificial intelligence and law 19*, 2–3 (2011), 187–231.

[9] Aucher, G., van Benthem, J., and Grossi, D. Modal logics of sabotage revisited. *J. Log. Comput. 28*, 2 (2018), 269–303.

[10] Baier, C., and Katoen, J.-P. *Principles of model checking*. MIT press, 2008.

[11] Bainbridge, D. Case: Computer assisted sentencing in magistrates' courts. In *BILETA Conference* (1990).

[12] Balbiani, P., Echahed, R., and Herzig, A. A dynamic logic for termgraph rewriting. In *ICGT* (2010), pp. 59–74.

[13] Beierle, C., Freund, B., Kern-Isberner, G., and Thimm, M. Using defeasible logic programming for argumentation-based decision support in private law. In *COMMA* (2010), vol. 216 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 87–98.

[14] Ben-Ari, M. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.

[15] Bench-Capon, T., and Prakken, H. Introducing the logic and law corner. *Journal of logic and computation 18*, 1 (2008), 1–12.

[16] Boella, G., Caro, L. D., Humphreys, L., Robaldo, L., Rossi, P., and van der Torre, L. Eunomos, a legal document and knowledge management system for the web to provide relevant, reliable and up-to-date information on the law. *Artif. Intell.*

Law 24, 3 (2016), 245–283.

[17] BOELLA, G., HUMPHREYS, L., MARTIN, M., ROSSI, P., AND VAN DER TORRE, L. Eunomos, a legal document and knowledge management system to build legal services. In *International Workshop on AI Approaches to the Complexity of Legal Systems* (2011), Springer, pp. 131–146.

[18] BONGIOVANNI, G., POSTEMA, G., ROTOLO, A., SARTOR, G., VALENTINI, C., AND WALTON, D., Eds. *Handbook of Legal Reasoning and Argumentation*. Springer, 2018.

[19] BORGES, F., BORGES, R., AND BOURCIER, D. A connectionist model to justify the reasoning of the judge. In *Legal Knowledge and Information Systems. Jurix 2002: The Fifteenth Annual Conference* (Amsterdam, 2002), T. Bench-Capon, A. Daskalopulu, and R. Winkels, Eds., IOS Press, pp. 113–122.

[20] BRYANT, R. E. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on 100*, 8 (1986), 677–691.

[21] BRYANT, R. E. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv. 24*, 3 (1992), 293–318.

[22] GABBAY, D., HORTY, J., PARENT, X., VAN DER MEYDEN, R., AND VAN DER TORRE, L., Eds. *Handbook of deontic logic and normative systems*. College Publication, 2013.

[23] GÄRDENFORS, P. *Knowledge in Flux (Modeling the Dynamics of Epistemic States)*. Bradford/MIT Press, Cambridge, Massachusetts, 1988.

[24] GIRARD, P., SELIGMAN, J., AND LIU, F. General dynamic dynamic logic. In *Advances in Modal Logic* (2012), pp. 239–260.

[25] GOVERNATORI, G., PADMANABHAN, V., ROTOLO, A., AND SATTAR, A. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL 17*, 3 (2009), 227–265.

[26] GOVERNATORI, G., AND ROTOLO, A. Changing legal systems: legal abrogations and annulments in defeasible logic. *Logic Journal of the IGPL 18*, 1 (2010), 157–194.

[27] GROSSI, D., AND ROTOLO, A. *A New Survey of Active Directions in Modern Logic*, vol. 30 of *Studies in Logic*. College Publications London, 2011, ch. Logic in the Law: A Concise Overview, pp. 251–274.

[28] HALPERN, J., HARPER, R., IMMERMAN, N., KOLAITIS, P., VARDI, M., AND VIANU, V. On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic 7*, 2 (2001), 213–236.

[29] JONES, A., AND SERGOT, M. Deontic logic in the representation of law: Towards a methodology. *Artificial Intelligence and Law 1*, 1 (1992), 45–64.

[30] LEITH, P. The rise and fall of the legal expert system. *European Journal of Law and Technology 1*, 1 (2010).

[31] MCCARTY, L. A language for legal discourse i. basic features. In *Proceedings of ICAIL* (1989), ACM, pp. 180–189.

[32] MOSS, L. S. Applied logic: A manifesto. In *Mathematical problems from applied logic I*. Springer, 2006, pp. 317–343.

[33] PRAKKEN, H. Formal systems for persuasion dialogue. *The Knowledge Engineering*

Review 21, 2 (2006), 163–188.

[34] Prakken, H., and Sartor, G. The role of logic in computational models of legal argument: a critical survey. *Computational Logic: Logic Programming and Beyond* (2002), 175–188.

[35] Rissland, E. L. *A companion to cognitive science*. Wiley-Blackwell, 1999, ch. Legal Reasoning, pp. 722–733.

[36] Satoh, K., Asai, K., Kogawa, T., Kubota, M., Nakamura, M., Nishigai, Y., Shirakawa, K., and Takano, C. Proleg: An implementation of the presupposed ultimate fact theory of japanese civil code by prolog technology. In *New Frontiers in Artificial Intelligence: JSAI-isAI 2010 Workshops* (2012), LNAI 6797, Springer, pp. 153–164.

[37] Sergot, M. J., Sadri, F., Kowalski, R., Kriwaczek, F., Hammond, P., and Cory, H. The british nationality act as a logic program. *Communications of the ACM 29*, 5 (1986), 370–386.

[38] Struillou, Y., Morin, M.-L., and Pécaut-Rivolier, L. *Le guide des élections professionnelles 2016-2017 et des désignations de représentants syndicaux dans l'entreprise*. No. 3. Dalloz / Guides Dalloz, 11 2015.

[39] Thireau, J.-L. *Introduction Historique au Droit*. Champs Université. Flammarion, 2009.

[40] van Benthem, J. *Exploring logical dynamics*. CSLI publications Stanford, 1996.

[41] van Benthem, J. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.

[42] Wagner, P. *Machine en Logique*. Presses Universitaires de France – PUF, 1998.

 Received 16 December 2018